日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　４月　１日

出願番号
Application Number: 特願２００３－０９７５７３

[ST. 10/C]: [ＪＰ２００３－０９７５７３]

願人
Applicant(s): 株式会社日立製作所

２００３年　９月１８日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００３－３０７６６９２

【書類名】　特許願

【整理番号】　K03002011A

【あて先】　特許庁長官殿

【国際特許分類】　G06F　9/45

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　本川　敬子

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　久島　伊知郎

【発明者】

【住所又は居所】　神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内

【氏名】　伊藤　信一

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社日立製作所

【代理人】

【識別番号】　100075096

【弁理士】

【氏名又は名称】　作田　康夫

【手数料の表示】

【予納台帳番号】　013088

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　階層メモリ向け最適化処理を備えたコンパイラおよびコード生成方法

【特許請求の範囲】

【請求項１】

計算機システムと協働することで、複数のメモリ階層を備えたアーキテクチャ上で実行するためのオブジェクトプログラムを生成するコンパイラであって、

対象プログラムが実行時に主としてどのメモリ階層上のデータを参照するかを指示するオプションまたは指示文を解釈するステップと、

上記指示されたメモリ階層向けの最適化処理を行うステップとを有することを特徴とするコンパイラ。

【請求項２】

請求項１のコンパイラであって、

前記メモリ階層向けの最適化処理として、メモリをアクセスする命令に対して指定されたメモリ階層に応じたメモリレイテンシを求め、求めたレイテンシに応じた最適化処理を行うことを特徴とするコンパイラ。

【請求項３】

請求項１のコンパイラであって、

メモリ階層向けの最適化処理として、メモリをアクセスする命令に対して指定されたメモリ階層に応じて、ループ交換またはループ展開またはループタイリングのループ変換方法を決定することを特徴とするコンパイラ。

【請求項４】

計算機システムおよび該計算機システムと協働して複数のメモリ階層を備えたアーキテクチャ上で実行するためのオブジェクトプログラムを生成するコンパイラにより遂行されるオブジェクトプログラム生成方法であって、

対象プログラムが実行時に主としてどのメモリ階層上のデータを参照するかを指示するオプションまたは指示文を解釈するステップと、

上記指示されたメモリ階層向けの最適化処理を行うステップとを有することを特徴とするコード生成方法。

【請求項５】

請求項４のコード生成方法であって、

前記メモリ階層向けの最適化処理として、メモリをアクセスする命令に対して指定されたメモリ階層に応じたメモリレイテンシを求め、求めたレイテンシに応じた最適化処理を行うことを特徴とするコード生成方法。

【請求項６】

請求項４のコード生成方法であって、

メモリ階層向けの最適化処理として、メモリをアクセスする命令に対して指定されたメモリ階層に応じて、ループ交換またはループ展開またはループタイリングのループ変換方法を決定することを特徴とするコード生成方法。

【請求項７】

複数のメモリ階層を備えたアーキテクチャ上で実行するプログラムに対して、対象プログラムが実行時に主としてどのメモリ階層上のデータを参照するかを指示するオプションまたは指示文を解釈することを特徴とするコンパイラ。

【請求項８】

請求項１のコンパイラを格納した記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、計算機の利用技術において、オブジェクトプログラムの実行時間を削減するコンパイル方法に関する。特に、複数のメモリ階層を備えたアーキテクチャ向けの最適化指示方法に関する。

【０００２】

【従来の技術】

マイクロプロセッサの速度向上に伴って主記憶アクセスのレイテンシが増大している。多くのプロセッサでは、主記憶よりもアクセスが速く比較的小容量のキャッシュ記憶を備えており、さらにキャッシュ記憶を、１次キャッシュ、2次キャッシュというように階層化することにより、なるべくレイテンシの小さいメモリ階層のデータをアクセスすることで、レイテンシの大きなメモリ階層のアクセ

ス回数を削減している。すなわちメモリアクセス命令は、１次キャッシュヒット時には短いレイテンシで１次キャッシュをアクセスし、１次キャッシュミス時には、２次キャッシュヒットならば2次キャッシュをアクセスし、全てのキャッシュ階層のミス時にのみ主記憶をアクセスする。

【０００３】

キャッシュミス時のレイテンシを隠す方法としては、ロード命令と、ロードしたデータを使用する命令の距離を十分離すような命令スケジューリングがあげられる。命令間を何サイクル離すかを決める基準としては、メモリレイテンシを用いる。

【０００４】

キャッシュミス時の主記憶参照のレイテンシを隠す別の手法として、例えば非特許文献１などに述べられているように、主記憶からキャッシュへデータを先行的に移動するプリフェッチ命令をプロセッサに用意し、コンパイラによってプログラム中にプリフェッチ命令を挿入する、ソフトウェア・プリフェッチ（プリフェッチ最適化）と呼ばれる方法がある。プリフェッチ命令を利用すれば、後続ループ繰り返しで参照するデータを予め主記憶からキャッシュへ移動しておきながら、同時に別の演算を行うことができ、主記憶参照のレイテンシを隠すことができる。

【０００５】

この方法では、ループ内のデータ参照に対してプリフェッチ命令を生成する場合、そのループの１回の繰り返しにかかる実行サイクル数Ｃを見積もり、データをメモリからキャッシュに移動するのに要するサイクル数（メモリレイテンシ）ＬをＣで割った値α＝CEIL(L/C)（CEILは小数点以下の切り上げを表す記号とする）を計算し、α回後の繰り返しで参照するデータを予めプリフェッチしておくことにより、Ｌサイクル後にそのデータを参照するときにはデータが既にキャッシュに到着しているのでキャッシュヒットとなり、プログラム実行が高速化される。１次キャッシュへのプリフェッチを行う場合、データが既に１次キャッシュ上にあればプリフェッチは不要である。また、データが２次キャッシュにデータがある場合にはＬは２次キャッシュから1次キャッシュへ移動するのに要するサ

イクル数、データが主記憶上にある場合には主記憶から１次キャッシュへ移動するのに要するサイクル数を使用した方がよい。しかし、通常、データがどのメモリ階層にあるかは不明なため、主記憶上にあると仮定した処理を行う。

【０００６】

データ局所性を高めるプログラム変換により、キャッシュミスの回数を削減する最適化も知られている。具体的な変換としては、ループタイリング、ループ交換、ループ展開があげられる。

【０００７】

ループタイリングとは、多重ループ内で参照されるデータがリユースを持つ場合に、一旦キャッシュ上にロードされたデータが他のデータの参照によってキャッシュから追い出される前に再度参照されるようにすることを目的としたループ変換である。ループタイリングについては、非特許文献２に記載されている。

【０００８】

メモリ参照パターンの最適化を目的としたループ交換やループ展開については、非特許文献３に記載されている。

【０００９】

以上で述べたようなレイテンシ隠蔽最適化やデータ局所化は、メモリ参照にかかるサイクル数や、キャッシュサイズといった、対象とするターゲットマシンの属性に依存する情報を必要とする。コンパイラでは通常、対象とするマシンの情報を内部情報として保持しているが、これらの情報をユーザが指示する方法もある。非特許文献４では'-mslatency=N'（Nは正定数）というオプション指定により、メモリからの読み出しにかかるサイクル数がNであることを指定できることが記載されている。非特許文献５では'-qcache'オプションにより階層キャッシュのレベル毎にキャッシュサイズやラインサイズ、連想数が指定できることが記載されている。

【００１０】

【非特許文献１】　Todd C. Mowry他：Design and Evaluation of Compiler Algorithm for Prefetching, Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992年

【非特許文献２】　Michael Edward Wolf, Improving Locality and Parallelism in Nested Loops', Technical Report: CSL-TR-92-538, 1992年

【非特許文献３】　Kevin Dowd, High Performance Computing', O'Reilly & Associates, Inc., 11.1節

【非特許文献４】　日立, (HI-UX/MPP for SR8000) Optimizing FORTRAN90 User's Guide', 6.2節

【非特許文献５】　IBM, 'XL Fortran for AIX User's Guide Version 7 Release 1', 第5章

【００１１】

【発明が解決しようとする課題】

従来の技術の項で述べたレイテンシ隠蔽最適化やデータ局所化は、プログラム実行時にデータがどのメモリ階層に存在するかにより最適化の方法は異なる。

【００１２】

例えば、命令スケジューリングにおいては、ロード命令と使用命令の間の距離が大きくなれば、使用レジスタ数が増大する。またプリフェッチ最適化においては、プリフェッチ命令のタイミングが早過ぎると、使用命令が実行される前に再びキャッシュから追い出されてしまう場合がある。したがって、これらの最適化で仮定するメモリレイテンシが大き過ぎると最適化の効果が十分に得られない。すなわちL2キャッシュヒットのデータに対して主記憶レイテンシを用いた最適化を適用すると、L2キャッシュレイテンシを用いた場合に比べて実行性能が低下する可能性がある。しかし、従来の技術ではロード命令の対象データがどのメモリ階層に存在するかが不明であるため、主記憶レイテンシを仮定せざるを得ない問題がある。

【００１３】

また、データ局所性最適化では、複雑なループ構造に変換したためにループ実行のオーバーヘッドが増加し実行性能が低下する可能性がある。ループ内で参照されるデータが主としてキャッシュミスを起こす場合にはキャッシュミス回数の削減により適用効果が得られるが、キャッシュヒット時にはキャッシュミス回数の削減効果はないため、適用しない方がよい。従来はデータがキャッシュヒット

するかどうかがわからないため、キャッシュヒット時にもループ変換を適用していた。このため実行性能が低下する場合があるという問題があった。

【００１４】

【課題を解決するための手段】

上記の課題を解決するために、本発明では、計算機システムと協働することで、複数のメモリ階層を備えたアーキテクチャ上で実行するためのオブジェクトプログラムを生成するコンパイラに、対象プログラムが実行時に主としてどのメモリ階層上のデータを参照するかを指示するオプションまたは指示文を解釈するステップと、上記指示されたメモリ階層向けの最適化処理を行うステップとを持たせた.

メモリ階層向けの最適化処理としては、メモリをアクセスする命令に対して指定されたメモリ階層に応じたメモリレイテンシを求め、求めたレイテンシに応じた最適化処理を行うステップ／またはメモリをアクセスする命令に対して指定されたメモリ階層に応じて、ループ交換またはループ展開またはループタイリングのループ変換方法を決定するステップを持たせた。

【００１５】

図１は本発明の概要を示す。図１においてソースプログラムTEST1.fの実行時に参照されるデータが主としてL2キャッシュデータであり、TEST2.fは主記憶データだとすると、これら２つのプログラムに対する最適化方法は異なるべきである。

【００１６】

しかし、従来はどちらも主記憶データを仮定した最適化を適用していた。プログラム内で参照されるデータが主としてアクセスするメモリ階層は、データサイズやループサイズなどによって決まる。これらはコンパイラの静的な解析では不明であることが多いためである。

【００１７】

本発明では、プログラム実行時にデータが主としてどのメモリ階層に属するかを指定する手段を設けることにより、コンパイラがメモリ階層の指定を解析し（１０１）、指定されたメモリ階層に応じた最適化を実施する（１０３、１０４、

１０５）ことで、より進んだ最適化が実施されたオブジェクトプログラムが生成できる。メモリ最適化では、指定されたメモリ階層のレイテンシやサイズなどの属性を用いてコード変換を実施する。図１の例ではTEST1.fに対してはL2データ向け最適化（１０４）を適用し、TEST2.fに対しては主記憶データ向け最適化（１０５）を適用する。

【００１８】

【発明の実施の形態】

以下、図面を用いて本発明の実施の形態について説明する。

【００１９】

図２は、本発明によるコンパイラが稼動する計算機システムの構成図である。この計算機システムは、ＣＰＵ２０１、ディスプレイ装置２０２、キーボード２０３、主記憶装置２０４、および外部記憶装置２０５より構成されている。キーボード２０３により、ユーザからのコンパイラ起動命令を受け付ける。コンパイラ終了メッセージやエラーメッセージは、ディスプレイ装置２０２に表示される。外部記憶装置２０５には、ソースプログラム２０６とオブジェクトプログラム２０７が格納される。主記憶装置２０４には、コンパイラ２０８、コンパイル過程で必要となる中間コード２０９とループ表２１０が格納される。コンパイル処理はＣＰＵ２０１がコンパイラプログラム２０８を実行することにより行われる。なお、ＣＰＵ２０１は内部にフェッチ／デコードユニット、実行ユニット等を含む処理ユニット２０１１からのアクセスレイテンシの異なるレベル１（Ｌ１）キャッシュ２０１２とレベル２（Ｌ２）キャッシュ２０１３を持ち、処理ユニットによる処理を高速に実行するためのメモリ階層を構成している。また、以降の説明では、コンパイラに記述されたコードを解釈して実行するＣＰＵ２０１による処理をコンパイラの処理／処理手順と言うものとする。

【００２０】

図３に、コンパイル時にメモリ階層指示を行うために、計算機システムが入力Ｊを受け付けるコンパイラ起動命令の例を示す。'f90'はFortranコンパイラの起動コマンドの例、test.fはソースプログラムの指定の例、'-O'はコンパイラオプションの一例である。本実施例ではメモリ階層指示オプションは'-data'で表し

、プログラム中のデータが主としてどのメモリ階層に存在するかを指定する。メモリ階層としては、L1キャッシュを表す'L1'、L2キャッシュを表す'L2'、主記憶を表す'MEM'を指定できる。'-data=L2'（３０１）では、データが主としてL2キャッシュに存在することの指示をしている。

【００２１】

図３の例ではコンパイラ起動命令中でメモリ階層指示を行う例を示すが、メモリ階層指示は、ソースプログラム中の指示文として記述することもできる。図４にメモリ階層指示が付加されたソースプログラムの例を示す。本例では、ループ単位に指示文を指定できるものとする。指示文４０１は'DO 10'のループ内のデータが主としてL1キャッシュヒットとなることの指示をしている。指示文４０２は'DO 20'のループ内のデータが主として主記憶をアクセスすることの指示をしている。指示文４０３は'DO 30'のループ内のデータが主としてL2キャッシュヒット（L1キャッシュミス）となることの指示をしている。

【００２２】

図５に、図２のシステムで稼動するコンパイラ２０８の処理手順を示す。コンパイラの処理は、構文解析５０１、メモリ最適化５０２、レジスタ割り付け５０３、コード生成５０４の順で行う。

【００２３】

構文解析５０１では、ソースプログラム２０６を入力として構文解析、およびループ解析を行い、中間コード２０９とループ表２１０を出力する。また本ステップにおいて、本発明の特徴であるメモリ階層指示文やメモリ階層指示オプションを解析し、ループ表に登録する。ループ表２１０については後で説明する。メモリ最適化５０２は、本発明の特徴となるステップであるので、後で詳細に説明する。レジスタ割り付け５０３では中間コード２０９の各ノードへのレジスタ割り付けを行う。コード生成５０４では、中間コード２０９をオブジェクトプログラム２０７に変換し、出力する。

【００２４】

図６にループ表２１０の例を示す。図には、ループを識別するためのループ番号６０１のほかは、本発明の特徴であるデータ指示フィールド６０２だけを示し

ている。ループ表は構文解析処理５０１の処理で生成されるが、その生成処理であるデータ指示フィールド設定の処理手順を図１３に示す。

【００２５】

本処理は、ループを順に辿り処理する。ステップ１３０１では、未処理のループがあるかどうかを判定し、あればステップ１３０２で未処理ループを１つ取り出す。ステップ１３０３ではコンパイル対象のソースプログラム中に図４に示したようなループへのメモリ階層指示である'data'指示文があるかどうかを調べる。あればステップ１３０７で'data'指示文のオペランドをループ表に設定する。図４の例ではDO10のループに対しては'L1'のように設定する。data指示文がないときにはステップ１３０４へ進み、コンパイラ起動命令にメモリ階層指示オプションである'-data'オプションの指定があるかどうかを調べる。オプション指定がある場合はステップ１３０５へ進み、オプションの値を設定する。オプション指定がなければステップ１３０６へ進み、「指定なし」を設定する。

【００２６】

以下、メモリ最適化５０２の詳細な例を示す。

【００２７】

最適化処理の第１の例はメモリ最適化５０２において命令スケジューリングを行うものとする。命令スケジューリングの処理手順を図７に示す。

【００２８】

ステップ７０１で、DAG(directed acyclic graph)を作成する。図８にA(I)*B(I)+C(I)に対応したDAGの例を示す。DAGの各ノードは中間コード上の命令に対応している。ノード間のエッジは、実行順序の制約を表す。例えばノード８０３のmul演算はノード８０１およびノード８０２のloadの結果をオペランドとするので、ノード８０１とノード８０２は、ノード８０３よりも先に実行しなければならない。この関係をノード８０１からノード８０３へのエッジ、およびノード８０２からノード８０３へのエッジで表す。

【００２９】

ステップ７０２で、ステップ７０１で作成したDAGのエッジ上にレイテンシを設定する。本ステップの処理手順の詳細を図９に示す。図９の各ステップについ

て説明する。

【００３０】

本処理は、DAG上のエッジを辿り、順に処理する。ステップ９０１では未処理のエッジがあるかどうかを判定し、なければ処理を終了する。あればステップ９０２へ進み、未処理のエッジを取り出して、処理対象とする。ステップ９０３では、エッジの始点ノードがロード命令かどうかを調べる。ロード命令でなければステップ９１０へ進み、始点ノードの演算に対応するレイテンシの値をエッジ上に設定して、このエッジの処理を終了し、ステップ９０１へ戻る。

【００３１】

ステップ９０４では、ノードの所属するループを調べ、ループ表によりdata指示を調べる。ステップ９０５では、data指示がL1かどうかを調べる。そうであればステップ９０９へ進み、L1キャッシュのレイテンシの値をエッジ上に設定してステップ９０１へ戻る。data指定がL1でなければステップ９０６へ進み、data指示がL2かどうかを調べる。そうであればステップ９０８へ進み、L2キャッシュのレイテンシの値をエッジ上に設定してステップ９０１へ戻る。そうでない場合、即ちdata指示がMEMまたは指定なしの場合にはステップ９０７へ進み、主記憶アクセスのレイテンシを設定してステップ９０１へ戻る。

【００３２】

図８は各ロードノードに対応するdata指示がL2で、L2レイテンシが10サイクルの場合の例を示している。

【００３３】

本実施例では、ロード命令の対象データがL1キャッシュ、L2キャッシュ、主記憶のいずれに存在しているかに応じてDAGエッジのレイテンシを設定することにより、ロード命令に対してより正確なレイテンシを仮定した命令スケジューリングを適用できるという効果がある。

【００３４】

次に、本発明によるメモリ最適化５０２において、プリフェッチ最適化を行う例を説明する。プリフェッチ最適化の処理手順を図１０に示す。

【００３５】

ステップ１００１では、未処理のループがあるかどうかを判定し、あればステップ１００２で未処理ループを１つ取り出す。ステップ１００３では取り出したループのdata指示をループ表２１０により調べ、data指示がL1ならば、ステップ１００１へ戻る。これは、このループのデータが既にL1キャッシュ上にあるため、プリフェッチ最適化の効果がないと判断したためである。ステップ１００４ではdata指示がL2かどうかを調べる。L2ならばステップ１００６へ進み、LにL2キャッシュのレイテンシを設定する。そうでない場合、即ちdata指示がMEMまたは指定なしの場合は、ステップ１００５へ進み、Lに主記憶レイテンシを設定する。

【００３６】

ステップ１００７ではループ繰り返しあたりの実行サイクル数を求め、Cに設定する。ステップ１００８では(L/C)を整数に切り上げた値をNに設定する。ステップ１００９ではNイタレーション後のプリフェッチコードを生成し、ループ内に挿入する。

【００３７】

本実施例ではL2キャッシュにあるかどうかに応じて主記憶レイテンシを設定することにより、より適切なプリフェッチ距離によるプリフェッチコードを生成する効果がある。また、L1キャッシュにデータがある場合には不要なプリフェッチコードを生成しない効果がある。

【００３８】

次に、本発明によるメモリ最適化５０２において、タイリングを行う例について説明する。タイリングの処理手順を図１１に示す。

【００３９】

ステップ１１０１では、未処理のループがあるかどうかを判定し、あればステップ１１０２で未処理ループを１つ取り出す。ステップ１１０３では取り出したループのdata指示をループ表２１０により調べ、data指示がL1ならば、ステップ１１０１へ戻る。これは、このループのデータが既にL1キャッシュ上にあるため、タイリングの効果がないと判断したためである。ステップ１１０４ではdata指示がL2かどうかを調べる。L2ならばステップ１１０６へ進み、ターゲットキャッ

シュをL1とする。そうでない場合、即ちdata指示がMEMまたは指定なしの場合は、ステップ１１０５へ進み、ターゲットキャッシュをL2とする。

【００４０】

ステップ１１０７では、タイリングの適用条件を調べ、適用対象かどうかを決定する。適用条件としては、多重ループであるか、依存条件を満たすか、タイリング効果があるかなどを調べる。適用条件の判定方法の詳細については、従来の技術の項で示した非特許文献２に記載されている。適用対象でなければこのループの処理を終了してステップ１１０１へ戻り、適用対象ならばステップ１１０８へ進む。

【００４１】

ステップ１１０８では、ステップ１１０５および１１０６で決定したターゲットキャッシュのキャッシュサイズ、連想度などに基づいてタイルサイズを決定する。タイルサイズを決定する方法については、従来の技術の項で示した非特許文献２に記載されている。

【００４２】

ステップ１１０９では、ステップ１１０８で決定したタイルサイズに従い、タイリング変換処理を行う。タイリング変換処理についても、従来の技術の項で示した非特許文献２に記載されている。

【００４３】

本実施例では、data指示に応じてターゲットキャッシュを決定することにより、より適切なタイルサイズによる変換を適用する効果がある。また、L1キャッシュにデータがある場合には不要なタイリング変換を適用しない効果がある。

【００４４】

次に、本発明によるメモリ最適化５０２において、ループ交換およびループ展開を行う例について説明する。処理手順を図１２に示す。

【００４５】

本実施例ではデータがL1キャッシュまたはL2キャッシュにある場合にはループ交換やループ展開の効果がないと考え、適用を中止し、データが主記憶にあると予想される場合のみに最適化を適用する。

【００４６】

ステップ１２０１では、未処理のループがあるかどうかを判定し、あればステップ１２０２で未処理ループを１つ取り出す。ステップ１２０３では取り出したループのdata指示をループ表２１０により調べ、data指示がL1またはL2ならば、ステップ１２０１へ戻る。これは、このループのデータが既にキャッシュ上にあるため、最適化の効果がないと判断したためである。ステップ１２０４ではループ交換の適用条件を調べ、適用対象ならばステップ１２０５でループ交換を適用する。ステップ１２０６ではループ展開の適用条件を調べ、適用対象ならばステップ１２０７でループ展開を適用する。

【００４７】

本実施例ではループ内のデータが主にキャッシュ上に存在する場合には、不要なループ交換やループ展開を適用しない効果がある。

【００４８】

メモリ最適化５０２において、以上説明したうちの全てまたはいくつかを組み合わせた最適化処理を行ってもよく、いずれか一つの最適化処理を行ってもよい。

【００４９】

【発明の効果】

本発明によれば、プログラム中のデータアクセスが主としてどのメモリ階層をアクセスするかを指定する手段を設けることにより、データがキャッシュ上に存在するときに効果のない最適化適用の抑止、指定されたメモリ階層に応じたメモリレイテンシを用いた命令スケジューリングやプリフェッチ最適化の適用、または／および指定されたメモリ階層に応じたターゲットキャッシュのキャッシュサイズ等のパラメタを用いたタイリングの適用が可能となり、プログラム実行の高速化が図れる。

【図面の簡単な説明】

【図１】　本発明の概要を示す図である。

【図２】　改良された最適化方法を実施するコンパイラが稼動する計算機システムの構成図。

【図３】　プログラム実行時に主としてアクセスするメモリ階層のコンパイラオプションによる指示の例を示す図。

【図４】　プログラム実行時に主としてアクセスするメモリ階層の指示文を含むソースプログラムの例を示す図。

【図５】　コンパイラの処理手順を示す図。

【図６】　ループ表の例を示す図。

【図７】　命令スケジューリングの処理手順を示す図。

【図８】　DAGの一例を示す図。

【図９】　DAGエッジのレイテンシ設定の処理手順を示す図。

【図１０】　プリフェッチの処理手順を示す図。

【図１１】　タイリングの処理手順を示す図。

【図１２】　ループ交換およびループ展開の処理手順を示す図。

【図１３】　データ指示フィールド設定の処理手順を示す図。

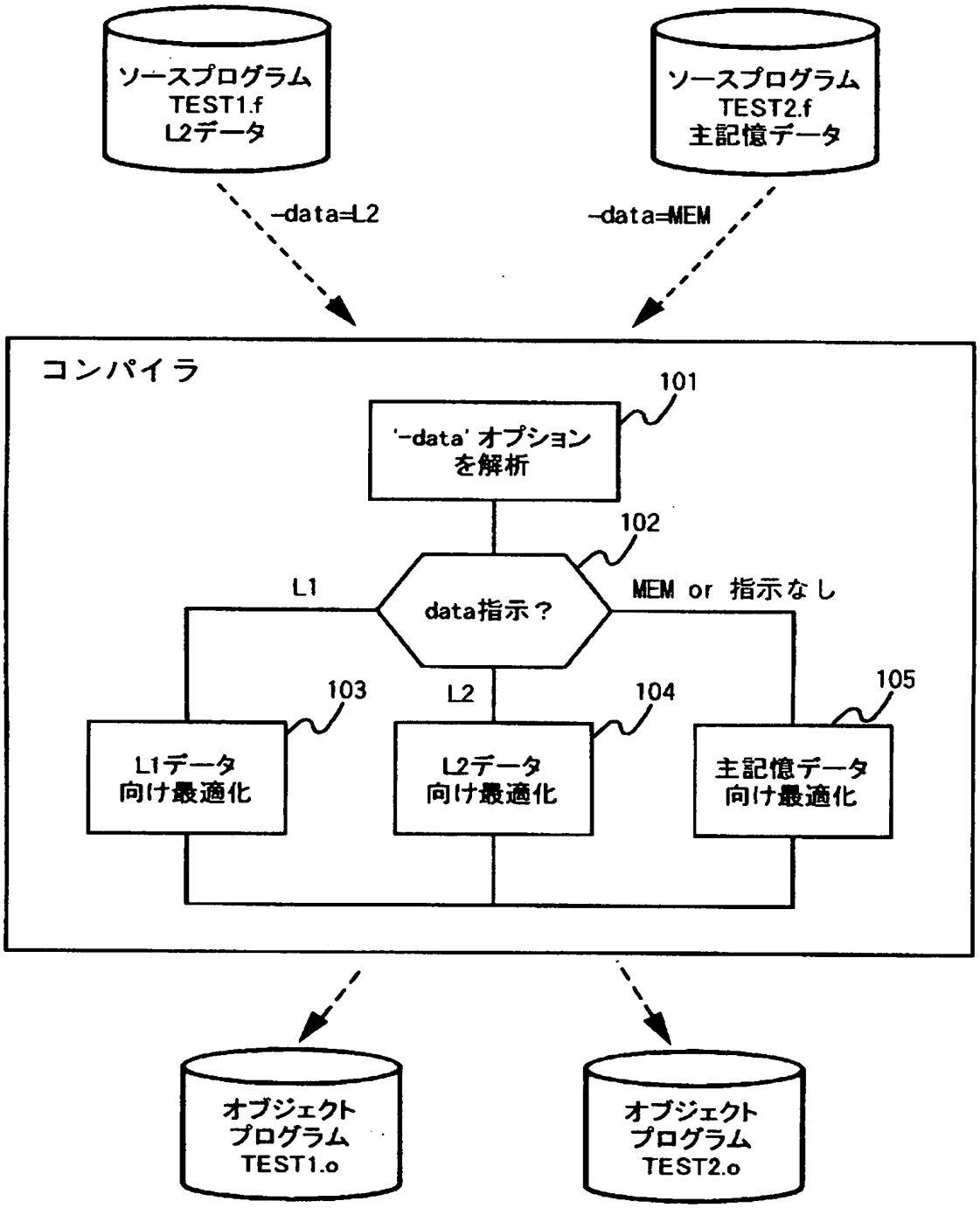
【符号の説明】

２０８・・・コンパイラ、５０２・・・メモリ最適化処理、１０３・・・L1データ向け最適化、１０４・・・L2データ向け最適化、１０５・・・主記憶データ向け最適化、３０１・・・メモリ階層指示オプション。

【書類名】　図面

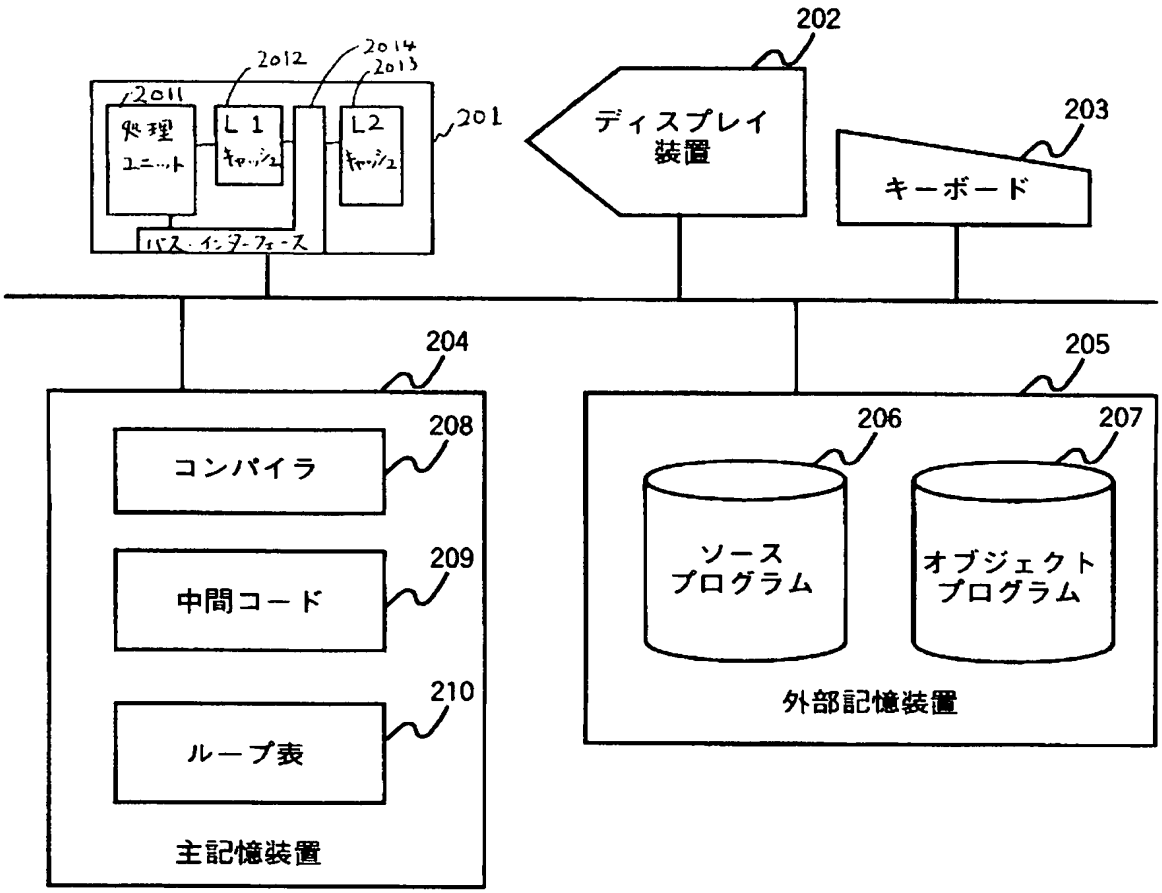
【図１】

本発明の概要（図１）



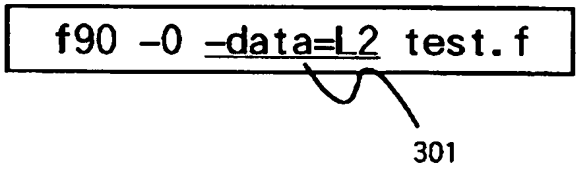
【図２】

コンパイラが稼動するシステムの構成図（図２）



【図３】

コンパイラオプションによる指示の例（図３）



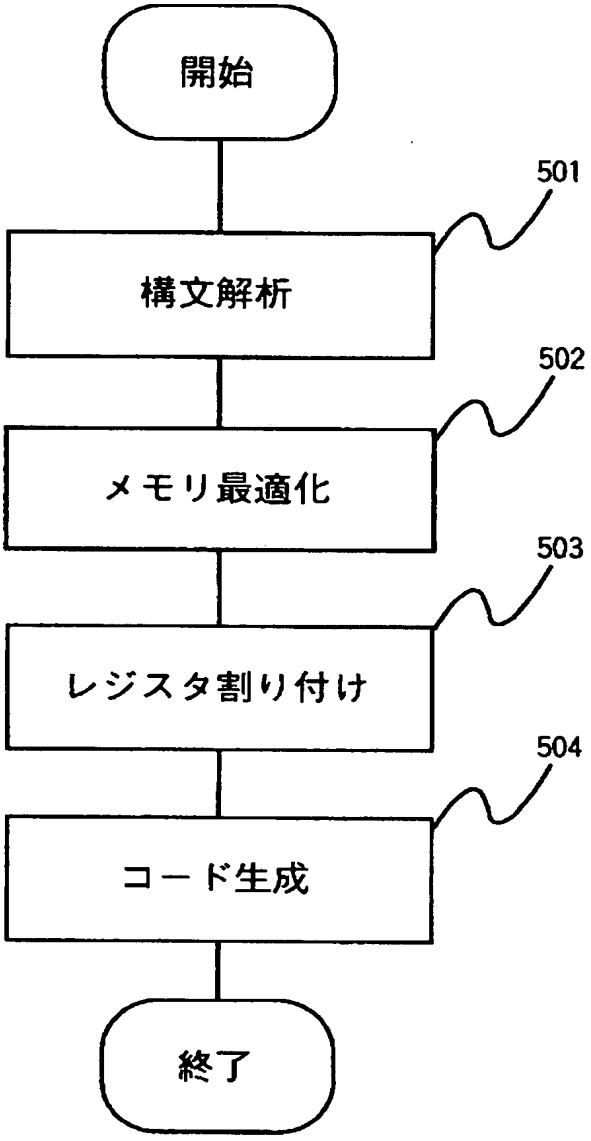
【図４】

ユーザー指示付きソースプログラムの例（図４）

```
      subroutine test1(A,B,C,D,E,F,N)
      real*8 A(N),B(N),C(N),D(N),E(N),F(N)
*option data(L1)                          401
      do 10 I=1,N
         A(I)=B(I)
 10 continue
*option data(MEM)                         402
      do 20 I=1,N
         C(I)=D(I)
 20 continue
*option data(L2)                          403
      do 30 I=1,N
         E(I)=F(I)
 30 continue
      end
```

【図５】

コンパイラの処理手順（図５）



【図６】

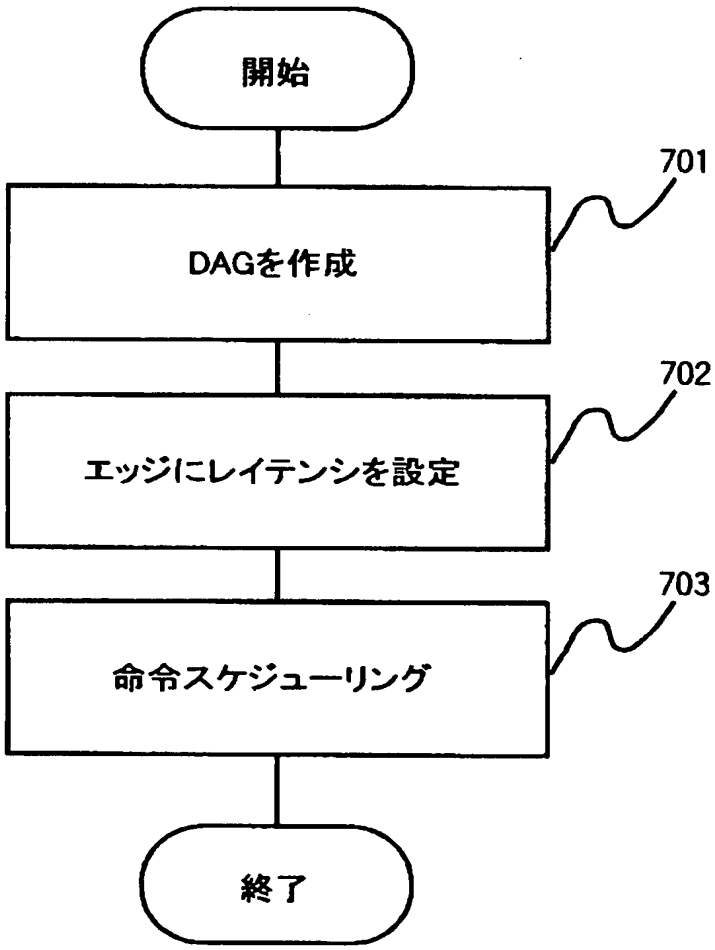
ループ表の例（図６）

601　602

ループ番号	data指示
1	L2
2	MEM
3	指定なし
4	L1

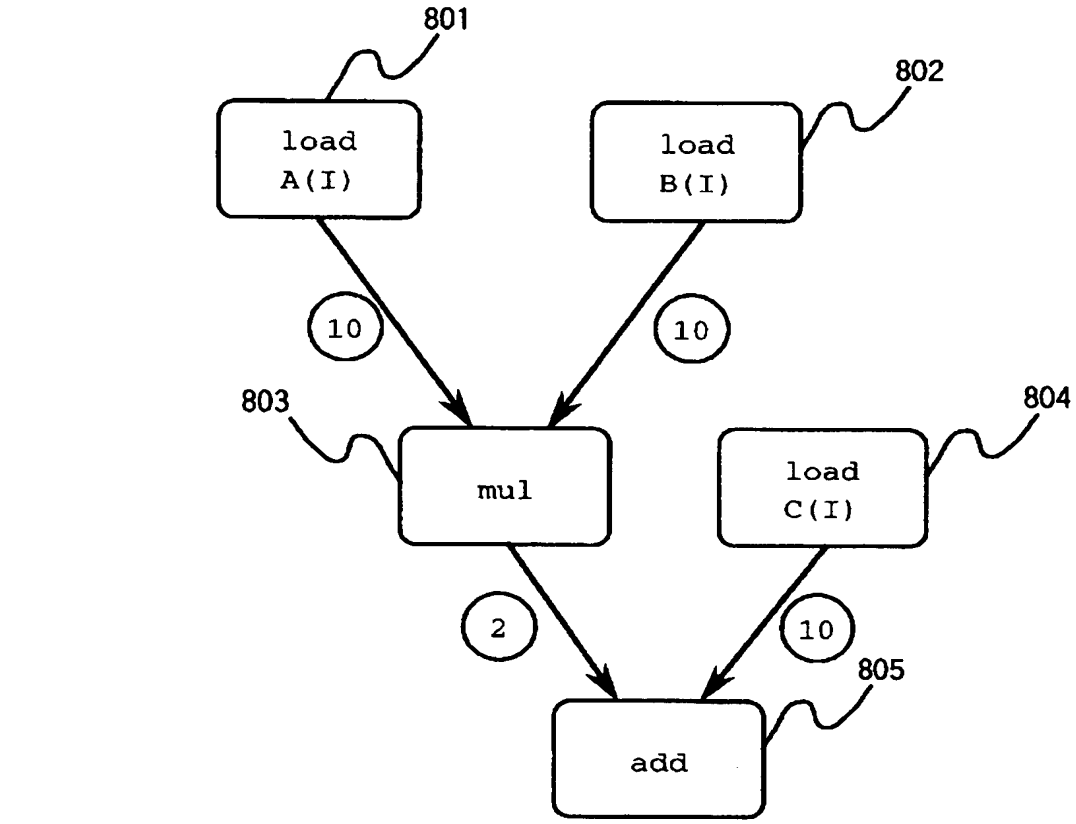
【図７】

命令スケジューリングの処理手順（図７）



【図８】

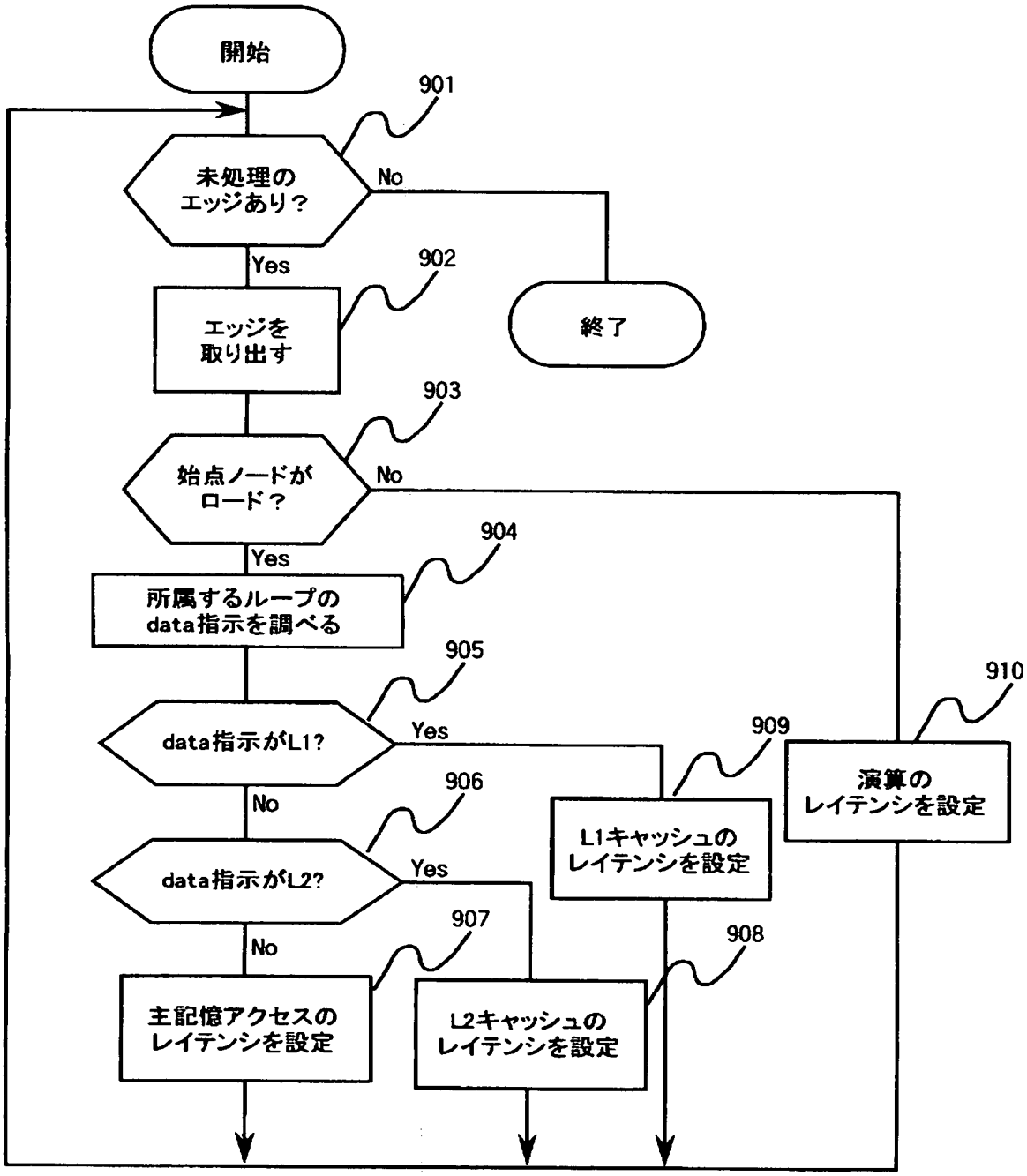
ＤＡＧの例（図８）



：レイテンシ

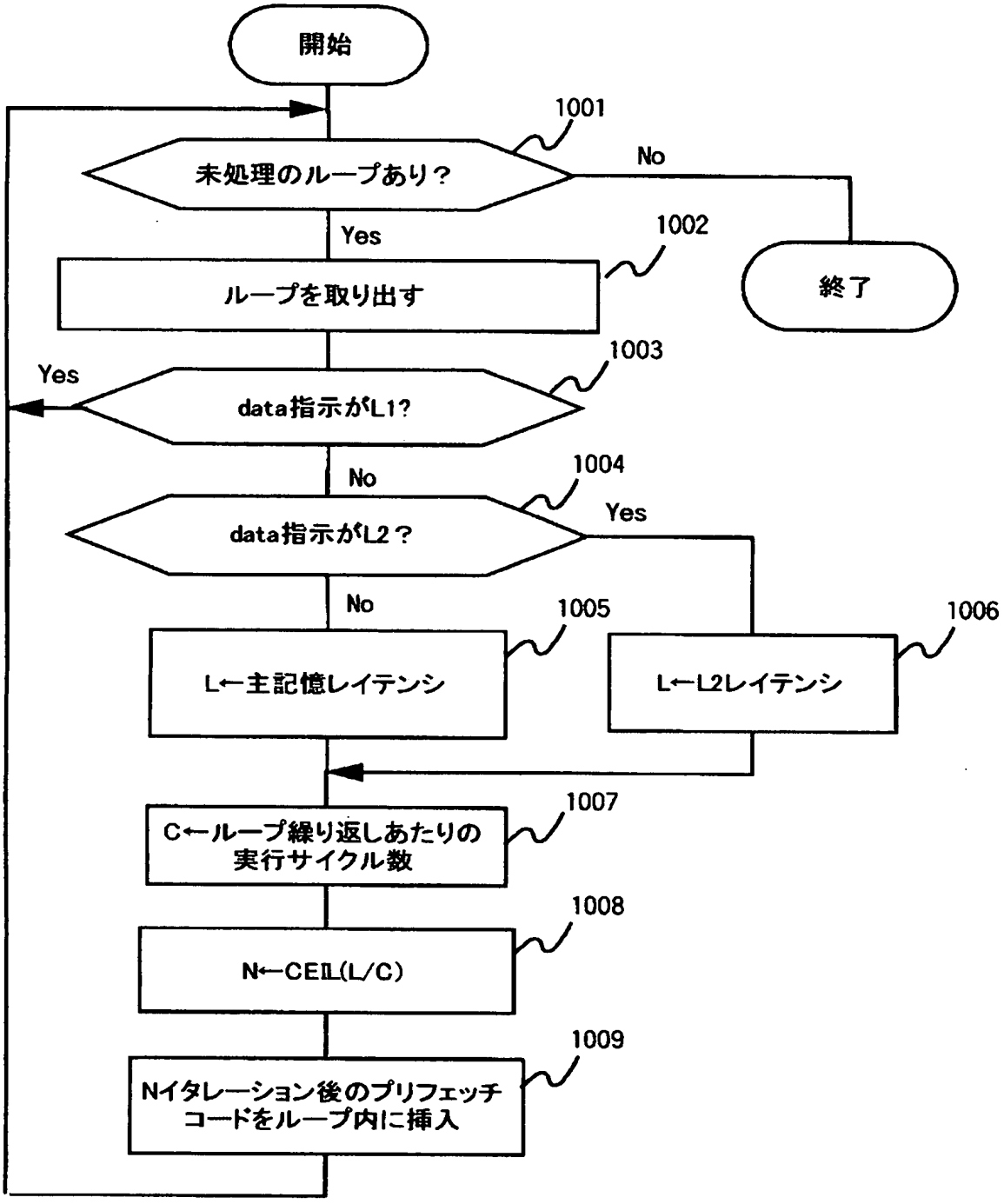
【図９】

ＤＡＧエッジのレイテンシ設定の処理手順（図９）



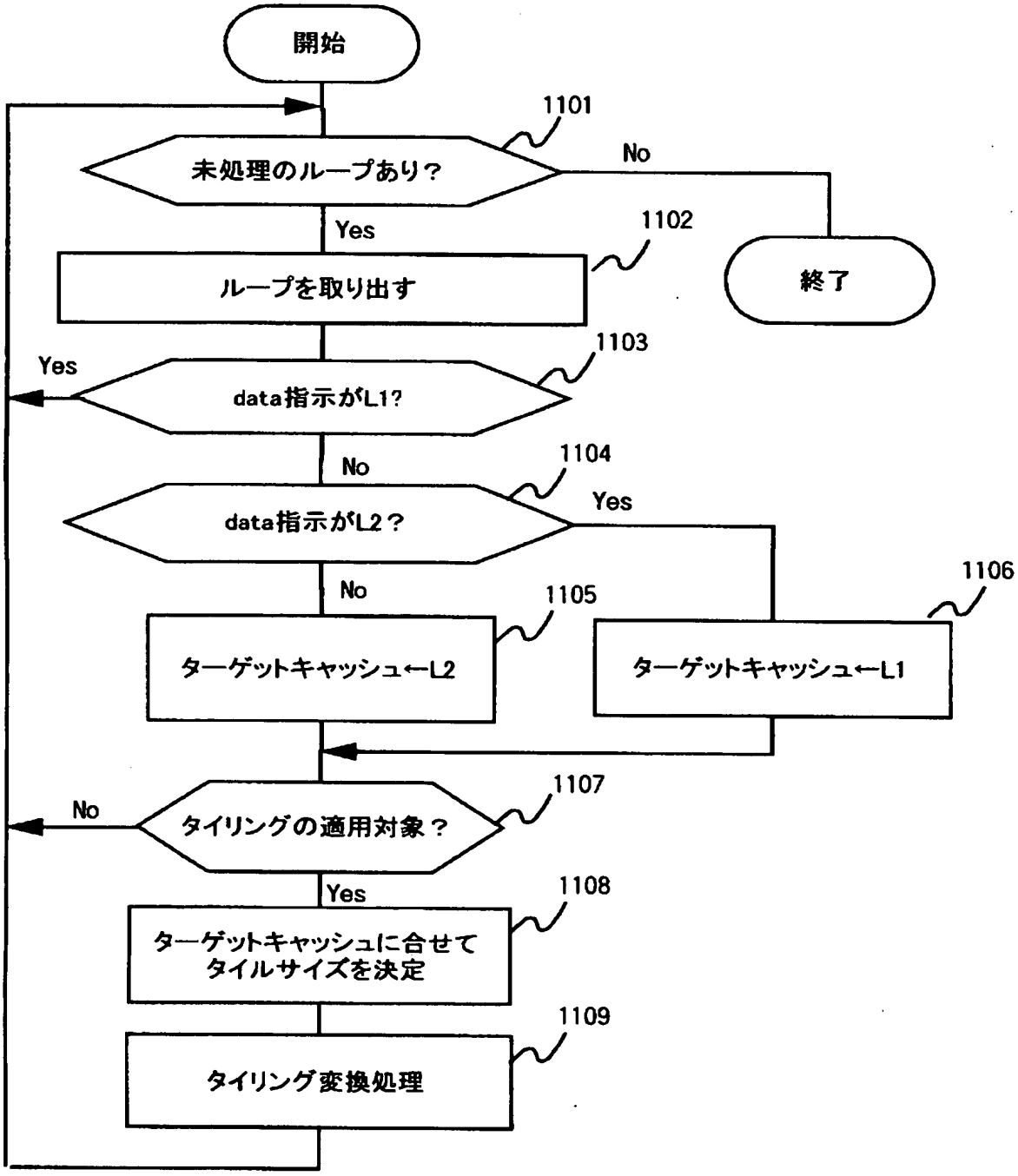
【図１０】

本発明によるプリフェッチの処理手順（図１０）



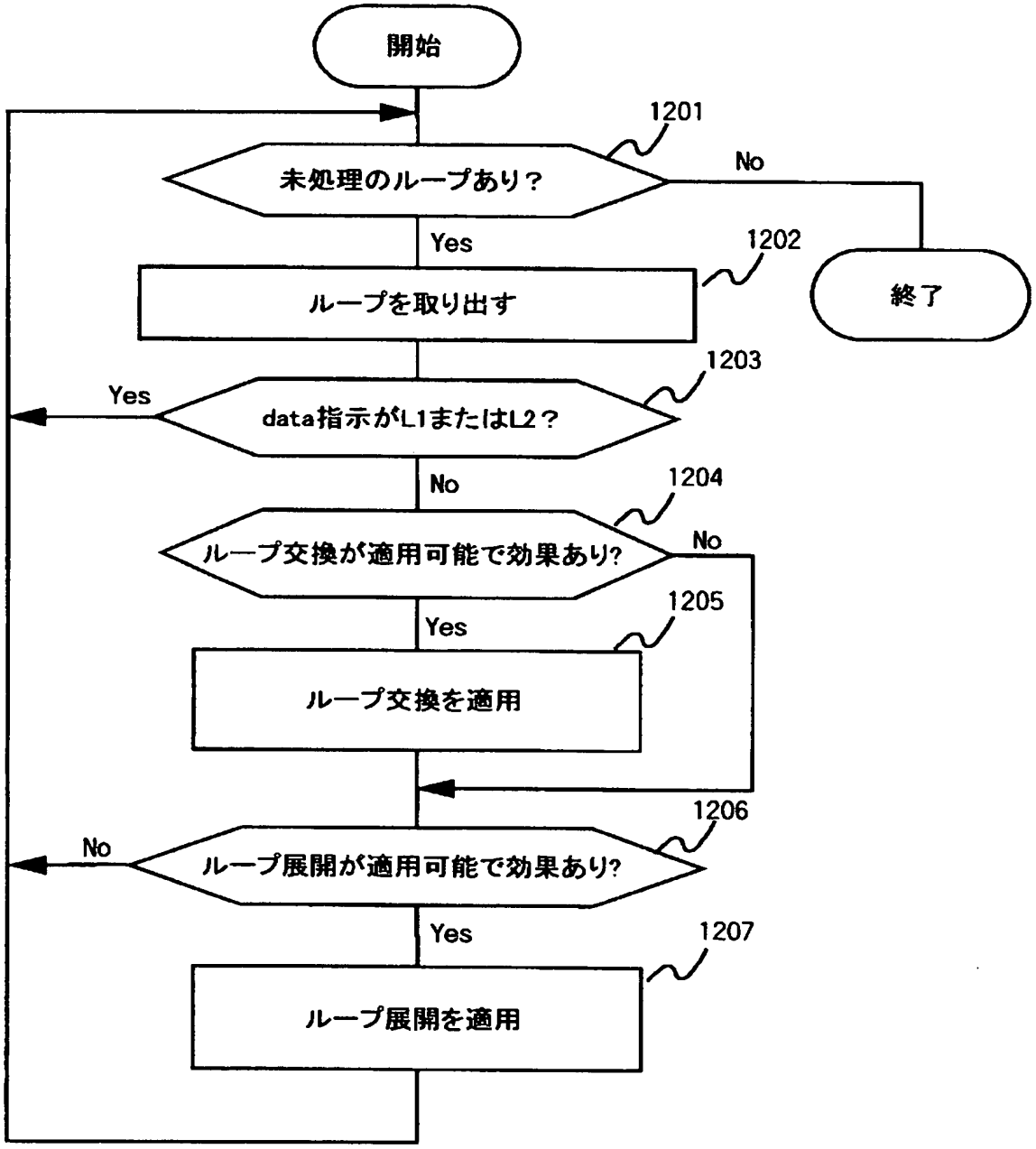
【図１１】

本発明によるタイリングの処理手順（図１１）



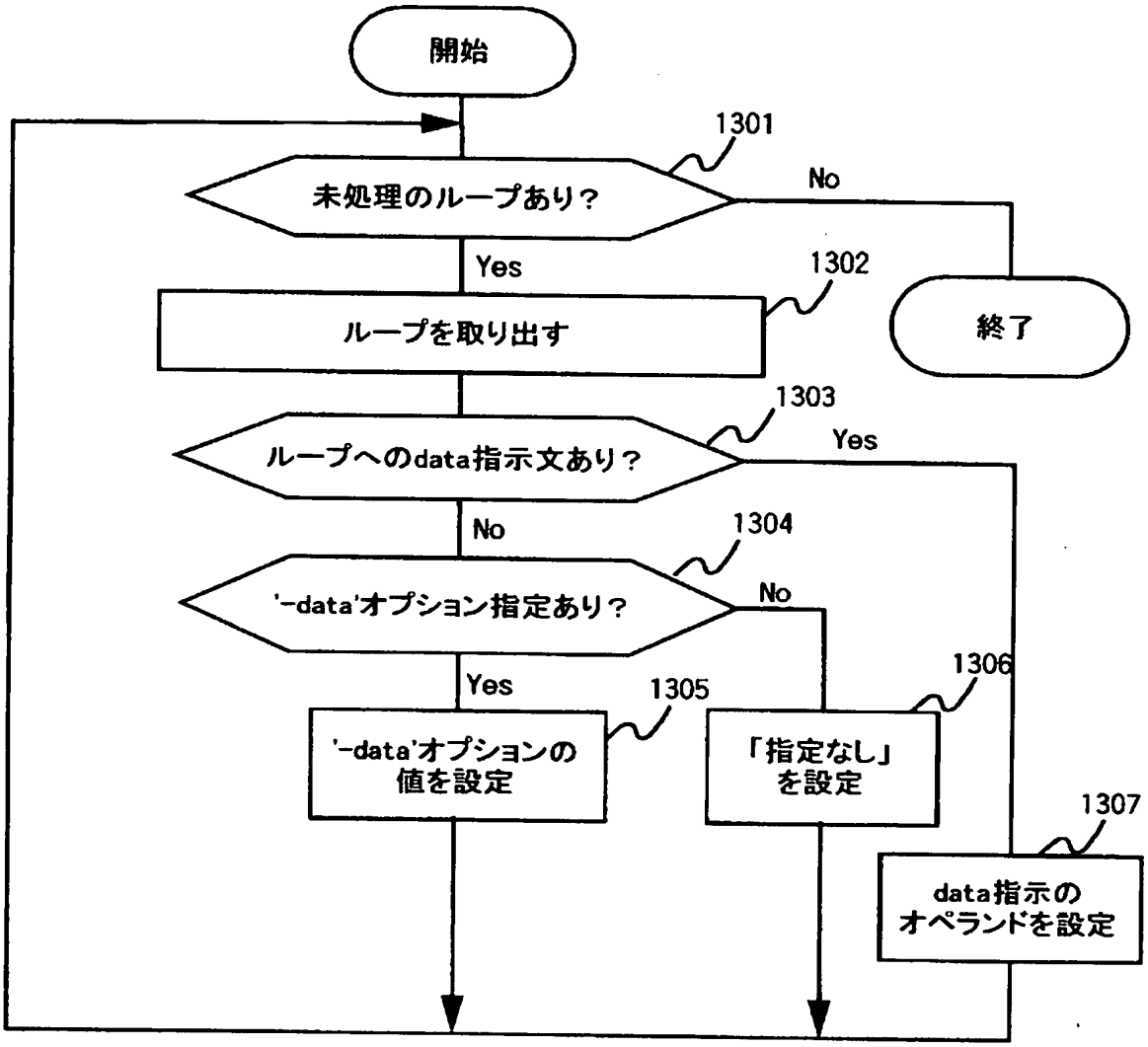
【図１２】

本発明によるループ交換・ループ展開の処理手順（図１２）



【図１３】

データ指示フィールド設定の処理手順（図１３）



【書類名】　要約書

【要約】

【課題】

プログラムが実行時に主としてアクセスするメモリ階層に応じて、異なる最適化方法を適用する方法を提供する。

【解決手段】

コンパイルオプション指示（３０１）またはプログラム中の指示文（４０１）を用いて、プログラムが主としてアクセスするメモリ階層をユーザが指定する。コンパイラ（２０８）ではメモリ階層指示を解析し（１０２）、メモリ階層に応じた最適化（１０３、１０４、１０５）を実施する。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２００３－０９７５７３
受付番号	５０３００５３９０１５
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　４月　２日

＜認定情報・付加情報＞

【提出日】　平成15年 4月 1日

次頁無

出　願　人　履　歴　情　報

識別番号　　［０００００５１０８］

１．変更年月日　　１９９０年　８月３１日
［変更理由］　　新規登録
住　所　　東京都千代田区神田駿河台４丁目６番地
氏　名　　株式会社日立製作所